06

# Tomando el control de la transacción

## Transcripción

[00:00] Hola. Vamos a avanzar con la mejoras en nuestro proyecto. Vamos a incrementar un poco la funcionalidad de creación de un producto y agregar una regla de negocio a ella. Nuestro depósito ahora tiene una limitación de solo poder soportar 50 cajas de un mismo producto por registro.

[00:16] Entonces si nosotros registramos un producto con cantidad mayor que 50, el código de BackEnd va a tener una lógica para dividir este procesamiento en dos operaciones de Insert. Vamos al código.

[00:31] Acá en el método guardar de productoController vamos a hacer lo siguiente. Vamos a realizar los cambios aquí para que podamos soportar este tipo de operación. Lo primero que vamos a hacer acá es extraer los valores del Map de producto para variables, porque la vamos a estar iterando, y vamos a crear una variable más acá para señalar un valor máximo. El campo cantidad puede tener por cada registro. El valor va a ser 50, como había dicho.

[00:59] Entonces acá el producto.get("Nombre"); voy a dar un "Ctrl + 1", extract to local variable, va a ser nombre acá, descripción= y el último va a ser la cantidad. Entonces abro acá la descripción y pongo cantidad. Voy a mover estos pedazos acá de código juntos para que queden acá arriba en el inicio de nuestra lógica.

[01:40] Y como había dicho voy a declarar una variable más del tipo Integer, que es el máximoCantidad con valor de 50. Ahí está. Ese es el inicio de la lógica. Antes de seguir con la lógica de negocio, haremos una refactorizaciór

acá en este código. Acá vamos a tomar esta parte del statement.set los valores, hasta el resultado y lo vamos a mover para un método, lo vamos a extraer para un método.

[02:12] Entonces voy a hacer un command o "Ctrl + 1" extract to method. Este método va a llamarse de ejecutaRegistro, donde va a estar recibiendo como parámetro el nombre, la descripción, la cantidad y el statement que nosotros inicializamos arriba para hacer su ejecución, la asignación de los parámetros y su ejecución.

[02:39] Miren una cosa acá. Me faltó cerrar la clase anterior la conexión acá del método guardar. Voy a compensar este fallo con el con.close ahora. Listo, compensado el fallo. Ahora vamos a nuestra lógica. Esta lógica debe ser ejecutada por lo menos una vez en el caso de que la cantidad de productos sea menor o igual que 50.

[03:02] Entonces nosotros podemos hacer acá un loop con el do while y la lógica va a quedar más o menos así: do, y acá vamos a hacer con esta lógica. Primero voy a cerrar porque me da un poco de cosa dejarlo abierto. Entonces do while, acá voy a poner la condición despuésm pero dentro del bloque de do while, nosotros vamos a hacer lo siguiente.

[03:27] Vamos a tomar un int cantidadParaGuardar, va a ser y acá voy a utilizar una librería. ¿Se acuerdan de las libs? La libs de MySQL, ahora va a utilizar la librería de matemática. Acá va a utilizar la Math.min. Yo quiero tomar el valor mínimo entre dos valores. ¿Qué valores son esos? La cantidad y el máximo de la cantidad.

[03:58] O sea, si es la cantidad tiene valor 100 y máximoCantidad es 50, el valor mínimo acá va a ser 50. Si el valor cantidad es 40 por ejemplo y el máximoCantidad es 50, el valor de cantidad para guardar va a ser 40. Entonces tomamos esta lógica acá, utilizamos una librería para abstraer esta lógica que

hoy vamos a estar haciendo con los ifs y en lugar de estar enviando la cantidad como parámetro, vamos a enviar el cantidadParaGuardar. Ahora sí.

[04:33] ¿Qué hacemos ahora con eso? Bueno, si vamos a esta ejecutando y tomando siempre el menor valor posible para ir guardando y pensando que el máximo que podemos guardar es 50, entonces si la cantidad es mayor que 50, la próxima vez que pasemos por este lazo, por este loop, nosotros tenemos que guardar lo restante.

[04:56] Entonces si estamos guardando 60 cucharas, por ejemplo, vamos en la primera pasada, a guardar 50 y en la segunda tenemos que guardar 10. ¿Cómo va a quedar la lógica acá?

[05:09] cantidad -= maximoCantidad. ¿Se entiende acá? Acá voy a estar haciendo una substracción del valor de cantidad del máximoCantidad. Entonces cada vez que pasemos, mientras la cantidad sea mayor que 50, vamos a estar sustrayendo 50. Perfecto. Acá, ¿cuál va a ser la condición para parar? Mientras cantidad>0 yo ejecuto mi loop. Voy a guardar acá el código, voy a levantar la aplicación y la vamos a probar. Tenemos la aplicación acá ejecutando.

[05:59] Lo que vamos a hacer primero es guardar 100 linternas. Acá voy a poner linternas con pilas recargables. Ahí está, 100 linternas. Guardar. Ahí está. Dos registros de 50. Pusimos 100 en el formulario y fueron guardadas 50 de cada uno en cada registro. Vamos a crear un registro de zapatillas, zapatillas de fútbol.

[06:37] Vamos a poner 40 zapatillas. Ahí fueron guardadas 40 también. Ahora vamos a hacer lo siguiente. Vamos acá a registrar algunas botellas y botellas de vidrio, vamos a poner 74 botellas de vidrio. Acá fue registrado con éxito. Mira, 50 guardadas primero, y después, 24 botellas que fue el restante.

[07:10] O sea, la lógica funciona bien y estamos agregando ahí cortado, cada 50 productos nosotros registramos uno nuevo en la tabla. ¿Pero ahora qué pasaría

si en el medio de la ejecución ocurre un error en la aplicación? ¿Todos los productos serían registrados? ¿Solo una parte o ninguna?

[07:34] Vamos a simular un error acá para ver qué puede pasar. Acá en el método ejecutaRegistro, nosotros vamos a poner una condición que antes de cualquier cosa, vamos a hacer un if(cantidad < 50) Si la cantidad es menor que 50 nosotros vamos a lanzar una RuntmeException, que dice ("Ocurrió un error"). Voy a bajar entonces la aplicación. Espera un poco.

[08:11] Me faltó acá, me faltó poner el new. Ahora sí, voy a bajar a la aplicación para ejecutarla una vez más. Ahora acá con la aplicación levantada vamos a registrar 60 platos. Entonces voy a poner acá platos de plástico, esos de fiesta, y la cantidad 60. Okay, guardar. No tenemos nada devuelto acá en la pantalla y en el log tenemos el error que pusimos para que se ejecute.

[08:46] Vamos a hacer lo siguiente porque no se recargó nada. Vamos a reiniciar la aplicación para ver qué pasa. Se reinició la aplicación y podemos ver acá que 50 de los platos fueron guardados y los 10 fueron perdidos. ¿Qué pasa acá? Lo que está pasando es que para cada ejecución del Insert la aplicación está abriendo una transacción para guardar los datos, devuelve el resultSet y cierra la transacción.

[09:13] El segundo intento va a pasar lo mismo. Se abre una nueva transacción, se insertan los datos, se devuelve un resultSet y se cierra la transacción. El tema es que cuando es lanzada una excepción, parte de la información es guardada y la otra parte es perdida. ¿Hay alguna forma de arreglar eso? Tomando el control de la transacción.

[09:36] Nosotros podemos tomar la responsabilidad del JDBC, que es manejar la transacción y hacerla nosotros para que la operación quede entera dentro de una sola transacción y garantizar que o se guarda todo o no guardamos nada. Aquí nosotros vamos a hacer lo siguiente.

[09:54] Acá en el método guardar, después que abrimos la conexión, nosotros recuperamos la conexión y luego de eso, nosotros vamos a setear para que ella no haga el commit de la transacción. El comando para tomar el control de la transacción va a ser, luego de abrir la conexión, vamos a hacer con.setAutocommit(false);

[10:14] Con eso nosotros configuramos acá que la conexión no va más a tener el control de la transacción y el que tiene el control de la transacción somos nosotros, y nosotros ahora podemos decir dónde el commit va a ser realizado. Ahora vamos a probar esta configuración para ver qué pasa con la base de datos.

[10:34] Nosotros vamos a reiniciar acá, voy a reiniciar la aplicación y ver cómo va a portarse la aplicación intentando agregar 60 zapatillas nuevas. Acá tenemos la aplicación levantada y vamos a agregar ahora. Ya tengo zapatillas de fútbol, ahora vamos a agregar nuevas zapatillas. Estas van a ser zapatillas de básquet. Ahí agregué 60 unidades. Puse para guardar.

[11:02] Hay un error en el log de la aplicación y no vemos nada todavía. Vamos a reiniciar la aplicación para ver si se guardó la información. Ahí levantó la aplicación otra vez y las zapatillas de básquet no fueron guardadas acá. El autoCommit funciona. Nosotros los pusimos como un false y no fue commiteada la transacción.

[11:23] Entonces aquí en el código ya podemos sacar esta situación de prueba que le hace el error y volver a intentar guardar nuevamente las 60 zapatillas. Voy a levantar la aplicación una vez más. Bueno, y acá en el formulario vamos a intentar nuevamente agregar las 60 zapatillas de básquet. Zapatillas de básquet, 60.

[11:47] Ahí hago clic en guardar. Tengo registrado con éxito. En el log también están 19 y 20 el id, pero no aparecieron acá en nuestro listado. ¿Que pasó con eso? En el log tenemos que está todo bien, pero no aparece acá en la tabla. Esto

pasa porque nosotros sacamos la responsabilidad del JDBC para concluir la transacción pero ahora nosotros no podemos registrar ningún producto.

[12:16] Resolvimos un problema pero ahora creamos otro. Este va a ser el tema para la próxima clase, en donde vamos a aprender cómo tratar esta situación y arreglar todo para que la aplicación vuelva a funcionar nuevamente. Nos vemos.